Use strongly-typed models

September 15, 2021 • Jan Cerman • 3 min read • .NET

Strongly typed models are programmatic representations of the content types in your Kontent project. They help you map the content retrieved from the Delivery API (which comes in the form of JSON objects) directly to a model you can use in your code.

Life without strongly typed models

When getting content directly from the Delivery API, you receive content items as JSON objects. Your application then needs to parse the JSON to display your content. For example, you access a *Headline* element of the example content item like this: `response.item.elements.headline.value`.

In a more complicated scenario, this approach becomes cumbersome. It forces you to remember the JSON structure of your content items and the codenames of your content elements.

To make this easier, use a Delivery SDK so that you can map the retrieved content items to their **strongly typed models**.

Use strongly typed models

This practice has several advantages:

- type safety during compile-time
- convenience (IntelliSense remembers content type properties for you)
- support of type-dependent functionalities (such as display templates in MVC)

The models are plain classes that don't have any attached behavior or dependency on an external framework. Each model corresponds to a content type in your Kontent project.

To create a content type's representation in code, you need a class with properties representing the individual content elements:

- Each property is mapped to its content element's codename either explicitly or using a naming convention. For example, codename `body_text` becomes property `bodyText`.
- Properties are usually typed using classes provided by the SDK.
- Depending on the SDK, simple content elements (like text and number) are sometimes represented using native types (like string and integer).

The following example is a strongly typed model of the *Homepage* content type.

C#

```
// This code was generated by a kontent-generators-net tool
// (see https://github.com/Kentico/kontent-generators-net).
//
// Changes to this file may cause incorrect behavior and will be lost if the code is
regenerated.
```

```
// For further modifications of the class, create a separate file with the partial class.

using System;
using System.Collections.Generic;
using Kentico.Kontent.Delivery.Abstractions;

namespace KenticoKontentModels
{
    public partial class Homepage
    {
        public const string Codename = "homepage";
        public const string HeadlineCodename = "headline";
        public const string BodyTextCodename = "body_text";
        public const string PictureCodename = "picture";

        public string Headline { get; set; }
        public IRichTextContent BodyText { get; set; }
        public IEnumerable<IAsset> Picture { get; set; }
        public IContentItemSystemAttributes System { get; set; }
    }
}
```

Generate models

We recommend that you generate the models using the .NET model generator.

To use the model-generator tool, follow the instructions in its README.

You can then use the generator to create strongly typed models from your Kontent project by providing your Project ID.

shell

```
dotnet tool install -g Kentico.Kontent.ModelGenerator
dotnet tool run KontentModelGenerator --projectid 8d20758c-d74c-4f59-ae04-ee928c0816b7 --withtypeprovider true --structuredmodel true
```

Retrieve strongly typed content

With your models defined and included in your application, you can use them when retrieving items from Kontent.

C#

```
// Tip: Find more about .NET SDKs at https://docs.kontent.ai/net
using Kentico.Kontent.Delivery;
using KenticoKontentModels;

// Creates an instance of the delivery client
// ProTip: Use DI for this in your apps https://docs.kontent.ai/net-register-client
IDeliveryClient client = DeliveryClientBuilder
```

```
        .WithProjectId("8d20758c-d74c-4f59-ae04-ee928c0816b7")
        .Build();

// Gets a content item by codename and maps it to the item's strongly typed model
IDeliveryItemResponse<Homepage> response = await client.GetItemAsync<Homepage>
("hello_caas_world");

var homepage = response.Item;
// Use homepage
// Console.WriteLine(homepage.Headline);
```

What's next?

In practice, you'll often have to define how your application resolves rich text elements containing assets, links, components, and other content items. This works a little differently for each platform, so we recommend that you have a look at the documentation of the SDK you are using:

- Using strongly typed models with Delivery .NET SDK
- Using strongly typed models with Delivery JavaScript/TypeScript SDK
- Using strongly typed models with Delivery Java SDK
- Using strongly typed models with Delivery PHP SDK
- Using strongly typed models with Delivery Swift SDK

For more advanced examples of using strongly typed models, see our sample applications.